A Project Report On

News Sentiment and Stock Movement Prediction: A Statistical NLP Framework

Submitted in partial fulfillment of the requirement for the 3^{rd} semester

Master of Science

in

STATISTICS

DEPARTMENT OF STATISTICS,

SCHOOL OF MATHEMATICAL SCIENCES,

KAVAYITRI BAHINABAI CHAUDHARI NORTH MAHARASHTRA UNIVERSITY

'A' Grade NAAC Re-accredited (4^{th} Cycle)

Umavi Nagar, Jalgaon - 425001 (M.S.) India

Submitted By

MARATHE HARSHAL BALKRUSHNA

Seat no. **366346**

Under the guidance of

Mr. MANOJ C. PATIL

Professor, dept. of Statistics , K.B.C N.M.U.

December 2024

KAVAYITRI BAHINABAI CHAUDHARI NORTH MAHARASHTRA UNIVERSITY

SCHOOL OF MATHEMATICAL SCIENCES

Re-accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade 4^{th} Cycle)

Umavi Nagar, Jalgaon - 425001 (M.S.) India

DEPARTMENT OF STATISTICS

CERTIFICATE

This is to certify that the project entitled **News Sentiment and Stock movement Prediction: A Statistical NLP Framework** is a bonafide work carried out by **HARSHAL BALKRUSHNA MARATHE [366346]** in partial fulfillment of 3rd semester, Master of Science in STATISTICS under Kavayitri Bahinabai Chaudhari University Narth Maharashtra University, Jalgaon during the year 2024-25.

Mr. Manoj C. Patil
(Project Guide)
Asst Prof. Dept. of Statistics, KBCNMU

Signature:.....

Acknowledgement

We are pleased to have successfully completed the project **NLP-Driven Stock Movement Prediction Using News Sentiment**. We thoroughly enjoyed the process of working on this project and gained a lot of knowledge doing so.

I would like to take this opportunity to express my gratitude to **Prof. (Mrs.) K. K. Kamalja** , Head of Department, Department of Statistics, for permitting me to utilize all the necessary facilities of the institution.

I am immensely grateful to my respected and learned guide, **Mr. Manoj C. Patil**, Assistant Professor, Dept. of Statistics for his valuable help and guidance. I am indebted to him for his invaluable guidance throughout the process and his useful inputs at all stages of the process.

I would also like to thank **Prof. Dr. R. L. Shinde** and **Prof. R. D. Koshti** for their help and guidance throughout the project process.

I also thank all the faculty and support staff of Department of Statistics, School of Mathematical Sciences. Without their support , this work would not have been possible.

Lastly, I would like to express my deep appreciation towards my classmates and my family for providing me with constant moral support and encouragement. They have stood by me in the most difficult of times.

Harshal Marathe (366346)

Abstract

The stock market is highly dynamic, with numerous factors influencing its movements, including economic indicators, market trends, and public sentiment. Traditional methods of stock market prediction primarily rely on historical price data, but recent advancements in Natural Language Processing (NLP) have shown that textual data, such as news articles, can offer valuable insights into market behavior. This project explores the application of NLP techniques to predict stock market movements by analyzing the sentiment of news articles related to the stock market.

The primary objective of this study is to develop a model that leverages NLP for sentiment analysis of stock news and combines it with historical stock prices to predict future market trends. We utilize a dataset consisting of stock news articles and corresponding stock prices, with the sentiment of the news articles quantified using the VADER sentiment analysis tool. An LSTM (Long Short-Term Memory) neural network model is employed to predict stock price movements based on the historical price data and sentiment scores.

The results of this project demonstrate the potential of integrating news sentiment into stock market prediction models. The model's performance is evaluated using standard metrics, including accuracy, precision, and recall, to assess its effectiveness in predicting stock price directions. The findings suggest that sentiment analysis, when combined with historical price data, can serve as a powerful tool for forecasting market trends, offering valuable insights for investors and market analysts.

This work contributes to the growing body of research in financial forecasting using NLP and provides a framework for future exploration in sentiment-driven stock market prediction models.

Contents

1 Introduction **9**
- 1.1 Background . . . 9
  - 1.1.1 Natural Language Processing (NLP) . . . 9
  - 1.1.2 Long Short-Term Memory (LSTM) . . . 10
- 1.2 Objective . . . 12
- 1.3 Scope of the Project . . . 13
- 1.4 Literature Review . . . 15
  - 1.4.1 Overview of Previous Research in Sentiment Analysis and Stock Prediction . . . 15
  - 1.4.2 Techniques Used for Sentiment Analysis . . . 16
  - 1.4.3 Findings Related to the Impact of News Sentiment on Stock Prices . . . 16

2 Statistical Analysis **18**
- 2.1 Data Preprocessing . . . 18
  - 2.1.1 Text Preprocessing . . . 18
  - 2.1.2 Text Representation . . . 20
- 2.2 Sentiment Analysis . . . 21
  - 2.2.1 Sentiment Analysis Techniques . . . 21
  - 2.2.2 Implementation . . . 22
- 2.3 Feature Engineering . . . 23
  - 2.3.1 Combining Features . . . 23
  - 2.3.2 MinMaxScaler for LSTM Model . . . 24
- 2.4 LSTM Architecture . . . 25
- 2.5 Challenges and Mitigation . . . 28

3 Application **29**
- 3.1 Data Collection . . . 29
- 3.2 LSTM Workflow for Stock Prediction . . . 30
- 3.3 Advantages of LSTM in Stock Prediction . . . 31
- 3.4 Evaluation . . . 31
- 3.5 Results . . . 33
  - 3.5.1 Results Visualization . . . 33
  - 3.5.2 Insights and Future Improvements . . . 33
  - 3.5.3 Challenges and Limitations . . . 34

4 Conclusion **35**
4.1 Limitations . . . 36
4.2 Future Directions . . . 36

A Appendix **38**
A.1 Data Extraction . . . 38
A.1.1 Stock Data Collection . . . 38
A.1.2 Stock News Collection . . . 38
A.1.3 Data Alignment . . . 39
A.2 Data Preprocessing . . . 39
A.2.1 Lower Casing . . . 39
A.2.2 Tokenization . . . 40
A.2.3 Stemming . . . 40
A.2.4 Lemmatization . . . 41
A.3 Text Representation . . . 41
A.3.1 One Hot Encoding . . . 41
A.3.2 Bag of Words . . . 42
A.3.3 TF-IDF . . . 43
A.3.4 Word2Vec . . . 43
A.3.5 Scaling of Features . . . 44
A.4 Modelling . . . 44
A.4.1 Calculation of Sentiment Score . . . 44
A.4.2 LSTM Model . . . 44
A.4.3 Evaluation . . . 45
A.4.4 Evaluation Metrics . . . 45
A.5 Hardware and Software Specifications . . . 46

List of Figures

1.1 NLP. 10
1.2 LSTM. 11

2.1 Text Preprocessing. 18

3.1 Confusion Matrix Heatmap. 34

Listings

A.1 Python code for stock data collection 38
A.2 Python code for stock news collection 38
A.3 Python code for aligning stock data and news data 39
A.4 Python code for lower casing 39
A.5 Python code for tokenization 40
A.6 Python code for stemming 40
A.7 Python code for lemmatization 41
A.8 Python code for One Hot Encoding 41
A.9 Python code for Bag of Words 42
A.10 Python code for TF-IDF 43
A.11 Python code for Word2Vec 43
A.12 Python code for scaling of features 44
A.13 Python code for calculation of sentiment score 44
A.14 Python code for LSTM model 45
A.15 Python code for evaluation 45
A.16 Python code for evaluation metrics 46

Chapter 1

Introduction

1.1 Background

1.1.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field of artificial intelligence that enables computers to understand, interpret, and respond to human language. It encompasses various techniques for transforming unstructured text data into a structured format that can be analyzed, which is critical for applications ranging from simple text analytics to sophisticated dialogue systems.

- **Importance of NLP in Data Science and Machine Learning Textual Data as a Resource**: With the increasing availability of textual data through news articles, social media, and other online platforms, NLP has become a cornerstone of data science. Text data can provide insights into customer preferences, public opinions, and market trends, making it an invaluable resource across industries.

  - **Core Tasks in NLP**: NLP includes various tasks, such as tokenization, stemming, and lemmatization, which prepare text for further analysis. Additionally, advanced tasks like sentiment analysis, named entity recognition, and topic modeling can extract specific insights and patterns.

  - **Advancements in NLP Models**: Over the years, NLP has evolved from basic keyword matching techniques to advanced deep learning models like RNNs, LSTMs, and transformer-based models (e.g., BERT, GPT). These models have vastly improved the accuracy of language processing tasks and enabled machines to understand context and semantics.

- **Applications of NLP in Finance**

  - **Market Sentiment Analysis** NLP is widely used in the financial sector to analyze news articles, earnings reports, and social media posts for sentiment. By understanding public sentiment toward a company or sector, investors can gauge market

expectations and adjust their investment strategies accordingly.

- **Algorithmic Trading**: Many algorithmic trading firms use NLP to monitor real-time news feeds, detect significant events, and automatically execute trades based on the predicted market impact.
- **Risk Management**: NLP helps in monitoring risk by tracking potential negative news events, such as legal issues, regulatory changes, or political developments, which could impact asset values.

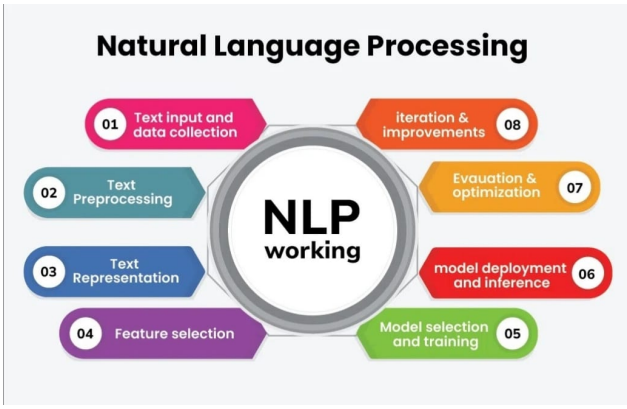

Figure 1.1: NLP.

1.1.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) that excels at processing and making predictions based on sequential data. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs were designed to overcome the vanishing gradient problem associated with traditional RNNs. This makes them particularly well-suited for tasks involving long-term dependencies, such as time-series prediction, natural language processing, and speech recognition.

1. **Why LSTM for Stock Prediction?** Stock prices and news sentiment scores are inherently time-dependent. Traditional Recurrent Neural Networks (RNNs) struggle with capturing long-term dependencies in such data. However, LSTMs offer several advantages that make them ideal for stock prediction:
   (a) **Sequential Nature of Data**: LSTMs excel at learning from ordered sequences, like past prices and sentiment data, which are crucial for understanding stock market dynamics.
   (b) **Memory Capability**: LSTMs utilize memory cells to retain important information while discarding irrelevant details. This allows them to understand the temporal context in stock movements, remembering the influence of past events on future prices.
   (c) **Handling Vanishing Gradients**: Traditional RNNs suffer from vanishing gradients

when dealing with long sequences. LSTMs overcome this issue with their specialized gating mechanisms, ensuring proper information flow within the network.

(d) **Integration of News Sentiment**: By incorporating sentiment scores derived from news articles as features, LSTMs can learn how news events influence stock prices over time, providing a more comprehensive understanding of market behavior.

2. **Applications of LSTMs in Stock Prediction**

- **Time-Series Forecasting:** Predicting stock prices or market indices using historical prices.
- **Sentiment Analysis Integration:** Incorporating news sentiment scores to predict the influence of public opinion on stock movements.
- **Event-Based Prediction:** Capturing the effects of major market events, such as earnings announcements or geopolitical developments.

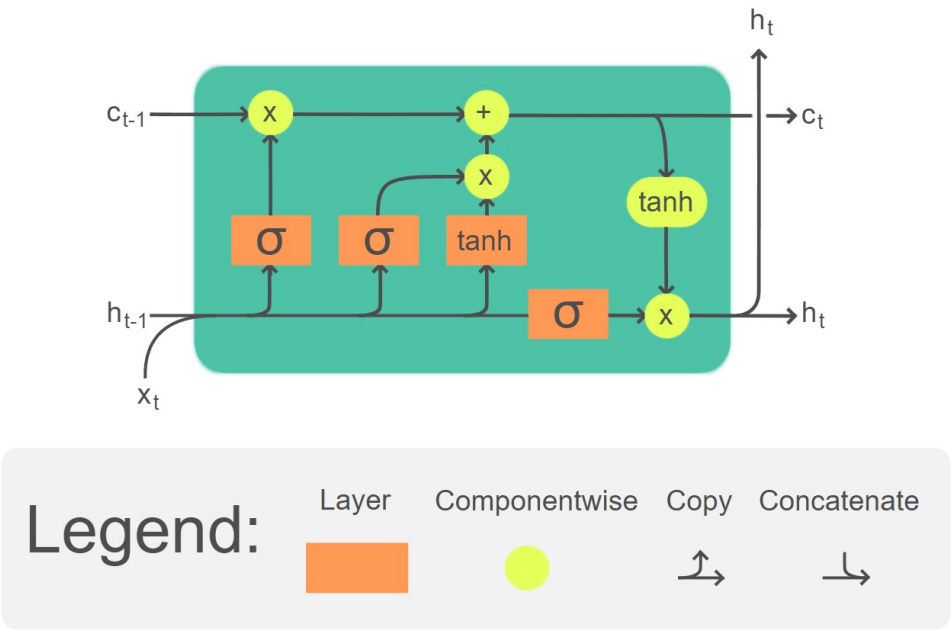

Figure 1.2: LSTM.

3. **Advantages of LSTMs**

- **Long-Term Dependency Handling:** Overcome the vanishing gradient problem to learn relationships spanning across time steps.
- **Flexibility:** Handle variable-length sequences and adapt to multimodal inputs.
- **Accuracy:** Proven to outperform traditional models (ARIMA, SVM) in capturing sequential patterns.

4. **Limitations of LSTMs**

- **Complexity:** Require significant computational resources and time for training.
- **Overfitting:** Sensitive to hyperparameter tuning and prone to overfitting on small datasets.
- **Interpretability:** Act as "black-box" models, making it difficult to interpret predictions.

1.2 Objective

Prediction of Stock Market Movements Using Sentiment Analysis
Predicting stock market movements through sentiment analysis of news articles has become a popular approach in financial analysis. This method leverages Natural Language Processing (NLP) techniques to gauge the emotional tone of news related to specific stocks, sectors, or the overall economy. By analyzing sentiment, traders and analysts aim to anticipate market reactions, making better-informed investment decisions. Here's an in-depth look at how sentiment analysis can be used to predict stock market movements:

1. Understanding the Role of News in Stock Market Movements

- Immediate Market Reactions: Financial markets are highly sensitive to news, as new information can quickly change investor perceptions and influence stock prices. Positive news often leads to buying pressure, while negative news can trigger selling.
- Public Sentiment as a Leading Indicator: News sentiment can reflect collective investor sentiment and expectations. Understanding the sentiment surrounding a company or industry can provide insights into how investors might react, which can, in turn, influence stock price movements.

2. Sentiment Analysis in Finance

- Textual Data Sources: News articles, press releases, earnings calls, and social media posts serve as rich sources of textual data. These sources offer real-time insights into public perception of financial events, which can drive market trends.
- Sentiment Scoring: Each news article or social media post is assigned a sentiment score based on its tone, often ranging from negative to positive. Sentiment analysis algorithms, like VADER (Valence Aware Dictionary and sEntiment Reasoner) or machine learning models, analyze these texts to assign sentiment scores.
- Aggregating Sentiment Scores: By aggregating sentiment scores over a specified time period, analysts can create a composite sentiment metric for a given stock or the market. For example, a high positive sentiment score for multiple articles on the same day could indicate optimistic market sentiment, potentially signaling a price increase.

3. Challenges in Sentiment-Based Stock Prediction

- Data Quality and Noise: Financial news can often be sensationalized or contain misleading information. Ensuring high-quality, reliable data is essential for accurate predictions.

- Timing and Latency: Sentiment analysis is not always immediate. For example, a news article's sentiment might impact a stock instantly or with a delay. Aligning this with the right stock price movements can be challenging.
- Sentiment Ambiguity: In financial contexts, words may carry different sentiments than in everyday language. For instance, "debt" in finance might carry a neutral tone, while it could be perceived negatively in a general context.
- Market Sentiment Shifts: Market sentiment can change rapidly, influenced by various factors such as economic policies, geopolitical events, and investor behavior. Capturing these shifts in real time requires sophisticated models and fast data processing capabilities.

4. Benefits and Implications

- Enhanced Decision-Making: By understanding the prevailing market sentiment, investors can make more informed decisions. For instance, if sentiment is largely negative for a stock despite good fundamentals, this could indicate a short-term opportunity.
- Risk Management: Sentiment analysis can serve as a risk management tool, allowing traders to detect potential downturns due to adverse news or detect excessive optimism that could lead to a bubble.
- Algorithmic Trading: Algorithmic traders can use sentiment-based models to trigger buy or sell orders. These models help exploit short-term opportunities that arise from sentiment shifts, making them valuable in high-frequency trading.

1.3 Scope of the Project

The scope of this project encompasses various aspects related to the use of Natural Language Processing (NLP) and sentiment analysis to predict stock market movements based on news articles. Below are the key elements that define the project scope:

1. **Specific Stock Focus** Target Stock: The project will focus on analyzing a specific stock (e.g., NVIDIA Corporation) to determine how news sentiment impacts its market performance. The choice of stock is significant due to its volatility and the influence of news events on investor behavior.
2. **Time Frame for Analysis**
    - **Historical Data Period**: The analysis will cover a defined historical period, such as the past 5 years, allowing for the examination of various market conditions and trends in sentiment and stock price movements.
    - **Granularity of Data**: Daily data will be used to align news articles with corresponding stock prices, providing a comprehensive view of how sentiment correlates with price changes over time.
3. **Preprocessing and Data Preparation**

- **Text Preprocessing**: The project will involve preprocessing steps, including tokenization, normalization, and sentiment scoring, to ensure accurate analysis of the news data.

4. **Feature Engineering**
   Integration of Features: The project will combine sentiment scores with traditional stock features, such as moving averages, volatility measures, and historical price changes, to create a comprehensive feature set for analysis.

5. **Modeling Techniques** Predictive Models: The project will explore various predictive modeling techniques, including:

   - **Machine Learning Models**: Such as Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting methods (e.g., XGBoost).

   - **Deep Learning Models**: LSTM (Long Short-Term Memory) networks will be employed to capture temporal dependencies in sentiment and stock price data.

   - **Model Evaluation**: The performance of different models will be evaluated using metrics such as RMSE (Root Mean Squared Error), accuracy, F1 score, and confusion matrices to assess their effectiveness.

6. **Comparative Analysis**

   - **Model Comparison**: A comparative analysis will be conducted to identify the most effective modeling techniques for predicting stock price movements based on news sentiment.

   - **Impact of Sentiment Analysis**: The project will evaluate the importance of sentiment in predicting market movements compared to traditional financial indicators, providing insights into the added value of sentiment analysis in financial forecasting.

7. **Limitations and Challenges**

   - **Acknowledgment of Limitations**: The project will address potential limitations, including biases in news coverage, market efficiency, and the unpredictability of stock price movements.

   - **Future Research Directions**: Suggestions for future research will be provided, highlighting areas for improvement and additional exploration within the domain of NLP and financial sentiment analysis.

1.4 Literature Review

1.4.1 Overview of Previous Research in Sentiment Analysis and Stock Prediction

Historical Context and Evolution

- Early Studies: Initial research focused on using simple keyword-based approaches to analyze news articles and financial reports. These methods often involved counting the frequency of positive and negative words to assess sentiment.
- Advancements in NLP: The advent of more sophisticated Natural Language Processing (NLP) techniques, including machine learning and deep learning, has revolutionized sentiment analysis. Models such as Support Vector Machines (SVM) and Random Forests have been employed to classify sentiment with higher accuracy.

Daniel Jurafsky and James H. Martin Jurafsky and Martin (2023)
The book Speech and Language Processing by Jurafsky and Martin is a foundational text in the fields of natural language processing (NLP), computational linguistics, and speech recognition. As the third edition draft comprehensively addresses modern advancements in NLP, it is particularly relevant for projects involving sentiment analysis, text representation, and predictive modeling, such as NLP-driven Stock Movement Prediction Using News Sentiment. Below, key aspects of the book are reviewed with a focus on their relevance to your project:

1. Foundations of Natural Language Processing The book begins with an introduction to the principles of language modeling, tokenization, and text preprocessing, which are crucial for preparing unstructured text data. It delves into methods such as n-grams, stemming, and lemmatization, laying the groundwork for transforming raw news data into analyzable formats. These preprocessing techniques directly support your goal of using past news for stock movement prediction by ensuring consistent and structured data representation.
2. Sentiment Analysis A significant section of the book explores sentiment analysis, particularly the use of machine learning methods to classify text polarity. Jurafsky and Martin provide detailed explanations of feature engineering, lexical resources (e.g., sentiment lexicons), and classification algorithms (Naive Bayes, SVMs, etc.). This aligns closely with the sentiment scoring aspect of your project, where news sentiment is analyzed and correlated with stock price trends.
3. Word Representations and Embeddings The authors address modern text representation techniques such as word embeddings (Word2Vec, GloVe) and contextual embeddings (ELMo, BERT). These techniques enhance the ability to capture semantic nuances in news articles, improving the model's understanding of market sentiment and its implications. Leveraging embeddings can help your LSTM model better associate linguistic patterns with stock price movements.
4. Sequence Modeling with RNNs and LSTMs Jurafsky and Martin discuss recurrent neural networks (RNNs) and their improved variant, long short-term memory networks (LSTMs),

which are central to your stock movement prediction model. Their explanation includes sequence modeling for tasks like text generation, time-series prediction, and speech recognition. These insights are invaluable for implementing an LSTM-based architecture to predict stock movements based on sequential news and price data.

1.4.2 Techniques Used for Sentiment Analysis

Sentiment Analysis Techniques

- Lexicon-Based Approaches: Tools like VADER (Valence Aware Dictionary and sEntiment Reasoner) have been widely used for sentiment scoring, particularly in financial contexts. Research shows that lexicon-based methods can effectively capture sentiment in news articles and social media posts, providing valuable insights into market trends.
- Machine Learning Models: Studies have demonstrated the effectiveness of machine learning models, such as Logistic Regression, SVM, and Naïve Bayes, in predicting stock price movements based on sentiment. These models leverage labeled datasets to train on historical sentiment and corresponding price changes.
- Deep Learning Models: Recent research has increasingly focused on deep learning approaches, such as Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). These models can capture temporal dependencies in sentiment data, leading to improved prediction accuracy. For instance, LSTMs have been utilized to analyze sequences of news articles and their impact on stock prices.

1.4.3 Findings Related to the Impact of News Sentiment on Stock Prices

Research has consistently demonstrated a significant relationship between news sentiment and stock prices. Below are key findings that highlight this connection:

1. Sentiment as a Predictor of Stock Returns
   Positive Sentiment Correlates with Price Increases: Numerous studies have found that positive news sentiment often leads to an increase in stock prices. For instance, a study by Zhang et al. (2018) Zhang et al. (2018) showed that stocks associated with positive news sentiment experienced an average price increase of 3-5% in the days following the publication of such news.
   Negative Sentiment Predicts Price Declines: Conversely, negative sentiment in news articles has been linked to a decline in stock prices. Research conducted by Tetlock (2007) Tetlock (2007) indicated that negative media coverage significantly predicted lower stock returns, particularly for firms with less favorable public perceptions.
2. Event-Driven Market Reactions
   Immediate Market Responses: Studies have shown that stocks often react quickly to significant news events. For example, Huang et al. (2022) Huang et al. (2022) found that stocks frequently experience sharp price movements within hours of negative news announcements, demonstrating the market's sensitivity to sentiment shifts.
   Earnings Announcements and Market Sentiment: Research has illustrated that investor

sentiment, influenced by news surrounding earnings announcements, can lead to volatility in stock prices. Stocks that receive favorable sentiment during earnings announcements typically see their prices rise, while those with negative sentiment tend to fall.

Chapter 2

Statistical Analysis

2.1 Data Preprocessing

Data preprocessing is a crucial step in preparing raw text data for input into RNN models. Proper preprocessing ensures that the data is in a suitable format and cleansed of unnecessary noise, which can significantly improve model performance. The following outlines the steps involved in data preprocessing for text generation using RNNs.

2.1.1 Text Preprocessing

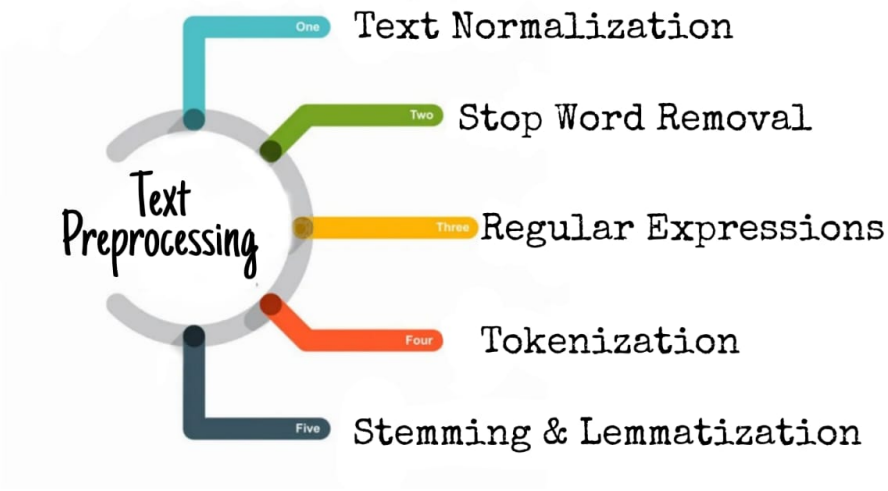

Figure 2.1: Text Preprocessing.

1. **Text Cleaning** Text cleaning involves removing unwanted elements and standardizing the text to ensure consistency across the dataset.

    - Lowercasing: Convert all text to lowercase to reduce the vocabulary size and ensure uniformity.
    - Removing Punctuation and Special Characters: Strip out punctuation marks, special characters, and numbers that do not contribute to text generation

    Appendix A.2.1

2. **Tokenization** Tokenization is the process of splitting the text into smaller units (tokens) such as words or characters. This is a key step for converting text into numerical form that can be processed by RNNs.

    - Word-Level Tokenization: Splits text into words, useful for models generating word sequences.
    - Character-Level Tokenization: Splits text into characters, which is useful for finer control in text generation, especially when handling unknown words or creative text generation.

    A.2.2

3. **Stemming and lemmatization** Stemming and lemmatization are two techniques used in natural language processing (NLP) to reduce words to their root form. While both aim to normalize text, they approach the task in different ways.

    Stemming

    - **Definition** : Stemming is a process of removing suffixes from words to reduce them to their root or stem form.
    - **Approach**: Stemming is typically rule-based, using a set of predefined rules to identify and remove suffixes.
    - Example: Stemming: "running" -> "run" Issue: Stemming can sometimes produce incorrect results, as it doesn't consider the word's context or part of speech. For example, "run" can also be the base form of "ran" and "runs."

    Appendix A.2.3

    Lemmatization

    - **Definition**: Lemmatization is a more sophisticated process that converts a word to its base form, considering its part of speech and morphological rules.
    - **Approach**: Lemmatization often involves using a dictionary or lexicon to look up the base form of a word.
    - Example: Lemmatization: "better" -> "good" Advantage: Lemmatization produces more accurate results than stemming, as it takes into account the word's context and meaning.

Appendix A.2.4

Key Differences:

Feature	Stemming	Lemmatization
Approach	Rule-based	Dictionary-based
Accuracy	Less accurate	More accurate
Context	Ignores context	Considers context
Output	Stem form	Base form

When to Use Which:

Stemming: Suitable for simple tasks where accuracy isn't critical, such as keyword extraction or information retrieval.
Lemmatization: Recommended for tasks where accuracy is important, such as sentiment analysis or text classification.

2.1.2 Text Representation

1. **Basic Text Representation Techniques**

   - **One Hot Encoding** One-hot encoding involves representing each word in a vocabulary as a unique binary vector. The vector length is equal to the size of the vocabulary, and each vector contains all zeros except for a single position, which is marked with a one. This position corresponds to the word's unique index in the vocabulary. Appendix A.3.1

   - **Bag-of-Words (BoW):** Represents text as a set of words, disregarding grammar and word order but keeping multiplicity. Each document is represented as a vector with word frequencies or binary values indicating the presence of words. Limitations include loss of context and inability to handle synonyms or polysemy effectively. Appendix A.3.2

   - **Term Frequency-Inverse Document Frequency (TF-IDF):** An enhancement over BoW by considering the importance of words based on their frequency across multiple documents.Helps down-weight common words and emphasize rarer, context-specific words.Despite improvements over BoW, TF-IDF still fails to capture the sequential nature of text, which is critical for RNNs. Appendix A.3.3

2. **Word Embeddings Word2Vec**: Introduced by Mikolov et al., Word2Vec creates dense vector representations of words using neural networks. Captures semantic relationships by training on context windows (CBOW or Skip-gram models). Words with similar meanings are placed closer in the vector space, enabling the model to understand relationships. Appendix A.3.4

2.2 Sentiment Analysis

2.2.1 Sentiment Analysis Techniques

VADER

The VADER (Valence Aware Dictionary and sEntiment Reasoner) tool is specifically designed for sentiment analysis of social media and news content. It's a lexicon and rule-based tool that's effective in capturing the sentiment intensity of text, especially short texts like news headlines or social media posts. In this project, VADER can be used to analyze news articles and measure their sentiment, which can then be used to predict stock price movements.
VADER provides four scores:

- Positive: The proportion of the text that is positive.
- Neutral: The proportion of the text that is neutral.
- Negative: The proportion of the text that is negative.
- Compound: The aggregated sentiment score, ranging from -1 to 1.

VADER Sentiment Score Algorithm

The following step-by-step algorithm outlines how the VADER (Valence Aware Dictionary and sEntiment Reasoner) calculates sentiment scores:

1. **Input**: A sentence or text string for sentiment analysis.
2. **Tokenization**:
    - Split the input text into individual words (tokens).
    - Preserve punctuation marks and special characters for accurate sentiment detection.
3. **Lexicon Score Look-Up**:
    - For each token, check if it exists in the pre-defined VADER lexicon.
    - **If Found**: Retrieve the associated sentiment score (positive, neutral, or negative).
    - **If Not Found**: Assign a score of 0 (neutral).
4. **Handle Special Cases**:
    - **Negation Words**: Identify tokens like "not," "no," or "never" that modify nearby sentiment-bearing words. Invert the sentiment of the following token(s) within a specific context window (e.g., 3 tokens).
    - **Punctuation**: Adjust the sentiment intensity for tokens followed by exclamation marks (!) or question marks (?).
    - **Capitalization**: If a token is in ALL CAPS, increase its sentiment intensity, assuming it conveys stronger emotion.

5. **Modify Sentiment for Degree Words (Intensity Modifiers)**:
    - Detect words like "very," "extremely," or "slightly" (intensifiers or dampeners).
    - Scale the sentiment score of the following token(s) up or down based on the degree word.
6. **Combine Sentiment Scores**:
    - Aggregate the scores of all tokens in the sentence:
        - Positive, Neutral, and Negative sentiment scores are summed separately.
7. **Compute Compound Sentiment Score**:
    - Calculate a **normalized compound score** using the formula:

$$\text{Compound Score} = \frac{\text{Sum of Weighted Sentiment Scores}}{\sqrt{\text{Sum of Squares of Sentiment Scores} + \epsilon}}$$

    - The compound score ranges from -1 (most negative) to $+1$ (most positive).
8. **Output**:
    - **Final Sentiment Scores**:
        - Positive, Neutral, and Negative scores as percentages.
        - Compound Score for overall sentiment evaluation.

Example: *Input Text*: "I absolutely love this product! It's amazing and works perfectly."

- **Steps**:
    - Tokenization: ["I," "absolutely," "love," "this," "product," "!," "It's," "amazing," "and," "works," "perfectly"].
    - Lexicon Look-Up: Retrieve scores for "love" (+3.0), "amazing" (+4.0), "perfectly" (+2.0).
    - Intensifiers: "absolutely" increases the score of "love" to +4.5.
    - Punctuation: Exclamation mark intensifies the score of the preceding token.
    - Combine Scores: Positive = 10.5, Negative = 0, Neutral = 2 (words without sentiment).
    - Compute Compound Score: Compound = +0.94.
- **Output**: Positive sentiment with a compound score of 0.94.

2.2.2 Implementation

In this implementation, we applied VADER (Valence Aware Dictionary and sEntiment Reasoner) to analyze the sentiment of news articles, which will later serve as input for predicting

stock price movements. The steps taken ensure accurate and efficient sentiment analysis of financial news articles, allowing for meaningful data extraction from text.

- 1. Data Preparation
  We start by loading a dataset containing financial news articles. This dataset includes columns such as content (the actual news text) and publishedAt (the publication date of the article). By loading this data into a DataFrame, we can efficiently apply transformations and calculations.

- 2. Sentiment Analyzer Initialization The SentimentIntensityAnalyzer from the nltk.sentiment.vader library is initialized. This tool uses a pre-trained lexicon specifically designed for analyzing the sentiment of short texts, such as news headlines and tweets, making it ideal for financial news sentiment analysis.

- 3. Sentiment Scoring A custom function is defined to extract the compound sentiment score for each news article. The compound score is a single, normalized value ranging from -1 (most negative) to +1 (most positive), summarizing the overall sentiment of the text.This function is then applied to each article in the dataset, generating a sentiment score. We add this score as a new column, sentiment_score, in the DataFrame.

- 4. Sentiment Categorization
  Based on the compound score, the sentiment is categorized as Positive,
  Neutral, or Negative:
  Scores ≥ 0.05 indicate a Positive sentiment.
  Scores ≤ -0.05 indicate a Negative sentiment.
  Scores between -0.05 and 0.05 are labeled Neutral.
  This categorization simplifies interpretation and allows for easier incorporation of sentiment data into models by transforming continuous sentiment scores into categorical labels.

Appendix A.4.1

2.3 Feature Engineering

2.3.1 Combining Features

Feature combining is a powerful technique to enhance your dataset's predictive capacity by creating meaningful relationships and interactions among features. It is essential to experiment with different combinations and validate their effectiveness through model evaluation, as not all combinations will necessarily improve performance. Proper feature engineering, including combining features, is a key step in the success of any predictive modeling project.

- Purpose: This section of code scales the Close prices and the sentiment_score features to a range between 0 and 1.

- Scaling is essential for LSTM models because: LSTMs are sensitive to the scale of the input data. If the input features have different ranges, the model might perform poorly or take

longer to converge during training.Scaling improves the optimization process, allowing the model to learn better and faster.

- MinMaxScaler: The MinMaxScaler from the sklearn.preprocessing module is used for this purpose. It transforms each feature by scaling it to a specified range (in this case, 0 to 1). The fit_transform method computes the minimum and maximum values of the features and scales them accordingly.

2.3.2 MinMaxScaler for LSTM Model

In the context of the *NLP-driven Stock Movement Prediction* project, the MinMaxScaler is used to normalize the data before feeding it into an LSTM model. This ensures that the input features are within a specified range, typically between 0 and 1, which improves the performance of the model.

1. **MinMax Scaling Overview**
   MinMax Scaling transforms the data so that each feature is rescaled to a predefined range, typically between 0 and 1. This transformation is essential for neural networks, particularly LSTM models, as they perform better when the input features are on a similar scale.

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

   Where:

   - X_{scaled} is the scaled value.
   - X is the original value of the feature.
   - X_{min} is the minimum value of the feature.
   - X_{max} is the maximum value of the feature.

2. **Why Use MinMaxScaler in LSTM Models?**

   - **Stability and Convergence:** Neural networks, including LSTM models, often struggle to learn effectively when input features have vastly different scales. By normalizing the data, the model can converge faster and improve accuracy.
   - **Time Series Data:** Stock prices often exhibit wide-ranging fluctuations, making it important to scale the data. This ensures that the model treats all features equally, preventing bias towards larger values such as stock prices.

3. **Applying MinMaxScaler in Stock Prediction** When using LSTM for stock prediction, you typically deal with time-series data like stock prices and sentiment scores. Here is how MinMaxScaler is applied to normalize these features:

4. **Normalizing Stock Prices and Sentiment Scores**

   - **Stock Prices:** MinMaxScaler can be used to scale the stock prices, ensuring that all values are transformed into a range between 0 and 1.

- **Sentiment Scores:** Sentiment scores, which can range from negative to positive values, are also normalized to ensure that they are on the same scale as the stock prices.

Scaling Data for LSTM

- Input Data: After scaling, you can split your data into input (X) and output (y) for training the LSTM model.
- Inverse Scaling: Once the model predicts stock movement, you can reverse the scaling to interpret the results on the original scale.

Appendix**??**

2.4 LSTM Architecture

The Long Short-Term Memory (LSTM) model used for stock movement prediction is designed to capture temporal dependencies in sequential data such as historical stock prices and news sentiment scores. Below, we describe the layers of the model in detail:

1. **Input Layer** The input layer receives data shaped as 3D tensors: **(batch size, sequence length, number of features)**
   - Batch size: Number of samples used per training iteration.
   - Sequence length: Number of past days' data used for predictions (e.g., 10 days).
   - Number of features: Features like previous closing prices and sentiment scores.
2. **LSTM Layer** The LSTM layer is the core component, processing the input data through memory cells:
   - Memory Cells : Each memory cell in the LSTM layer has three gates to control information flow:
     (a) Forget Gate (ft): Decides what information to discard from the cell state.

$$ft = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where:

ft : Forget gate output
h_{t-1} : Previous hidden state
x_t : Current input
σ : Sigmoid activation function

(b) Input Gate (it): Determines which new information to store in the cell state.

$$it = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where:

it : Input gate output

$\tilde{C}_t$: Candidate memory

(c) Output Gate (ot): Controls what information from the cell state to output.

$$ot = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

where:

ot : Output gate output

h_t : Output (hidden state)

Cell State (Ct): Updated at each step to store relevant information.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

3. **LSTM Layer 1**
   - **Units:** 50 LSTM cells.
   - **Output Shape:** $(sequence\_length, 50)$.
   - **Configuration:**
     - *Return Sequences:* True (to pass output to the next LSTM layer).
     - *Activation Function:* Internal gating uses sigmoid and tanh activations.
     - *Dropout Rate:* 20% to reduce overfitting.
4. **LSTM Layer 2**
   - **Units:** 50 LSTM cells.
   - **Output Shape:** (50).
   - **Configuration:**
     - *Return Sequences:* False (as this is the last LSTM layer, it outputs the final hidden state).
     - *Dropout Rate:* 20%.

5. **Dropout Layers** Dropout layers are added between LSTM layers to prevent overfitting by randomly setting a fraction of LSTM units to zero during training.

6. **Dense Layer** A fully connected layer that maps the LSTM outputs to the target variable (stock price or movement).

    - **Units:** 1 neuron.
    - **Activation Function:** Linear (suitable for regression tasks like stock price prediction).
    - **Output:** Predicted stock price for the next day.

7. **Output Layer** Outputs the predicted stock price or movement.

    - Regression: Linear activation for predicting exact stock prices.
    - Classification: Sigmoid or Softmax activation for predicting upward/downward movement.

Summary of Model Architecture
The overall structure of the LSTM model is summarized as follows:

```
Layer (type)                 Output Shape              Param #
=================================================================
LSTM_1 (LSTM)                (None, sequence_length, 50) n_params
Dropout_1 (Dropout)          (None, sequence_length, 50) 0
LSTM_2 (LSTM)                (None, 50)                 n_params
Dropout_2 (Dropout)          (None, 50)                 0
Dense_1 (Dense)              (None, 1)                  51
=================================================================
Total Parameters: x,xxx
Trainable Parameters: x,xxx
Non-trainable Parameters: 0
=================================================================
```

Visual Representation of Model Architecture
The LSTM architecture can be visualized as:

Input Layer: (sequence_length, num_features)
↓
LSTM Layer 1 (50 Units, Return Sequences = True)
↓
Dropout Layer 1 (Rate = 0.2)
↓
LSTM Layer 2 (50 Units, Return Sequences = False)
↓
Dropout Layer 2 (Rate = 0.2)
↓
Dense Output Layer (1 Unit)

2.5 Challenges and Mitigation

While LSTMs are powerful, they face certain challenges:

- **Overfitting:**
  - Using dropout layers and early stopping can help mitigate overfitting.
- **Training Time:**
  - Training LSTMs can be computationally expensive. Using GPUs and reducing model complexity can accelerate training.
- **Data Limitations:**
  - Limited availability of high-quality data can hinder model performance. Augmenting data with related stock indices or sectors can be beneficial.

Chapter 3

Application

3.1 Data Collection

Stock Data

1. **Collection Method** To collect historical stock data, there are several popular APIs and libraries available that provide reliable and accessible financial data, including stock prices. In this project, you can choose from sources such as Yahoo Finance, Alpha Vantage, or other platforms that offer APIs. Here's an outline of the stock data collection process:
   Yahoo Finance API
   Library: yfinance
   Description: Yahoo Finance is one of the most widely used sources for stock data. The yfinance library in Python offers a simple way to download historical stock price data for a specific ticker symbol.
   Data Features: The stock data will typically include features such as:
   - Date : which includes the timeline.
   - Closing Price: The final price at which the stock was traded on a given day.

   Appendix A.1.1

News Data

1. **Collection Method** You can collect historical news data using the NewsAPI library in Python. NewsAPI is a popular service that provides access to a wide variety of news articles from multiple sources, with the ability to filter by keyword, date, language, and more. For this Project I have taken the data from Kaggle Website "Reliance News from 2017 to 2021 ".
2. **Time Period Selection** For this project, the selected time period is critical as it should capture significant events, market fluctuations, and trends relevant to Reliance stock prices.
   - Time Period: January 1, 2017, to October 30, 2022.

This period includes major events affecting the tech industry, economic fluctuations, and Reliances's corporate milestones.

Appendix A.1.2

Data Alignment

- **Purpose of Data Alignment**: Data alignment aims to ensure that the different data sources used in the prediction model, such as historical stock prices, news sentiment scores, and potentially other market indicators, are properly synchronized. This is essential for accurate modeling and forecasting, as discrepancies in time or structure can lead to inaccurate predictions and analyses.
- **Sources of Data:** Historical Stock Prices: This includes daily closing prices, opening prices, high and low values, and trading volumes of the stocks (e.g., RELIANCE). News Data: This includes timestamps of articles, sentiment scores derived from the articles, and possibly other features like keyword frequency or category.
- **Steps for Data Alignment**
  - Timestamp Standardization: Ensure that all datasets have a common time format (e.g., UTC) to facilitate accurate alignment. Convert timestamps to a uniform timezone if necessary.
  - 'Resampling or Interpolating Data: Stock price data is often available on a daily basis, while news articles may be available at irregular intervals. You may need to resample the stock prices to match the frequency of news articles or vice versa.
  - Merging Datasets: Combine the datasets into a single DataFrame using the date/-time column as the key. Ensure that this merge preserves all relevant columns, including sentiment scores and stock prices.

Appendix A.1.3

3.2 LSTM Workflow for Stock Prediction

1. **Input Data** The LSTM model requires the following input data:
   - **Historical Stock Prices:** Previous day's closing prices, technical indicators (e.g., moving averages, RSI, Bollinger Bands), and other relevant historical data.
   - **News Sentiment Scores:** Aggregated daily sentiment scores derived from news articles related to the stock.

   The target variable can be:
   - **Price Movement (Up/Down):** For classification tasks.
2. **Preprocessing** Before training the LSTM model, data preprocessing is essential:

- **Normalization:** Features are scaled to a specific range (e.g., 0 to 1) using techniques like Min-Max scaling or standardization to ensure stable training.
- **Sequence Generation:** The input data is reshaped into sequences of fixed length (e.g., 10 days). Each sequence represents a time window of historical data used to predict the future value.

3. **Training** The LSTM model is trained using the following steps:
   - **Loss Function:**
     - Classification: Binary Cross-Entropy loss is used to assess the model's ability to classify price movements.
   - **Validation Split:**
     - A portion of the training data is set aside for validation to monitor the model's performance and prevent overfitting.
4. **Prediction** Once the model is trained, it can be used to predict future stock movements. Given a sequence of recent historical data, the model generates predictions for the next time step.

Appendix A.4.2

3.3 Advantages of LSTM in Stock Prediction

LSTMs offer several advantages for stock prediction:

- **Temporal Awareness:** LSTMs can effectively capture long-term dependencies in time series data, allowing them to model the impact of past events on future outcomes.
- **Feature Flexibility:** They can handle multiple input features, including historical prices, technical indicators, and news sentiment scores.
- **Robustness to Noise:** LSTMs are relatively robust to noise in financial data, enabling them to capture underlying trends and patterns.

3.4 Evaluation

Model Evaluation The model's performance was evaluated using the following metrics and techniques:

Evaluation Metrics for Classification

1. **Accuracy** Accuracy measures the ratio of correctly predicted samples to the total number of samples. It is given by:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy is best used for balanced datasets but may not be ideal for imbalanced datasets.

2. **Precision, Recall, and F1-Score** These metrics are particularly suitable for imbalanced datasets.
   Precision: Precision represents the proportion of true positive predictions out of all positive predictions. It is calculated as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

   Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive samples. It is expressed as:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

   F1-Score: The F1-Score is the harmonic mean of precision and recall. It is computed as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. **Confusion Matrix** The confusion matrix provides a summary of prediction results in a tabular format, which includes:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

   - **True Positive (TP):** Correctly predicted positive samples.
   - **False Positive (FP):** Incorrectly predicted as positive.
   - **True Negative (TN):** Correctly predicted negative samples.
   - **False Negative (FN):** Incorrectly predicted as negative.

   Appendix A.4.4

Evaluation Results: The model achieved the following : **Confusion Matrix and Classification Report**

1. **Confusion Matrix** The confusion matrix summarizes the performance of the classification model. It is represented as:

$$\begin{bmatrix} 37 & 74 \\ 36 & 124 \end{bmatrix}$$

   Where:

   - 37: True Positives (TP)
   - 74: False Positives (FP)
   - 36: False Negatives (FN)

- 124: True Negatives (TN)

2. **Classification Report** The detailed classification report includes precision, recall, F1-score, and support for each class:

	precision	recall	f1-score	support
0	0.51	0.33	0.40	111
1	0.63	0.78	0.69	160
accuracy			0.59	271
macro avg	0.57	0.55	0.55	271
weighted avg	0.58	0.59	0.57	271

3. **Overall Metrics** The overall evaluation metrics for the classification model are as follows:

   - **Accuracy:** 0.594
   - **Precision:** 0.626
   - **Recall:** 0.775
   - **F1-Score:** 0.693

3.5 Results

3.5.1 Results Visualization

3.5.2 Insights and Future Improvements

1. Insights:

   - The model performed well in predicting stock price trends but struggled during highly volatile periods.
   - News sentiment scores contributed significantly to improving prediction accuracy.

2. Future Improvements:

   - Incorporate more advanced sentiment analysis techniques (e.g., transformer-based models like BERT) for better sentiment scoring.
   - Use additional features, such as macroeconomic indicators, trading volume, and global market indices.
   - Experiment with hybrid models (e.g., combining CNNs with LSTMs) for improved feature extraction.

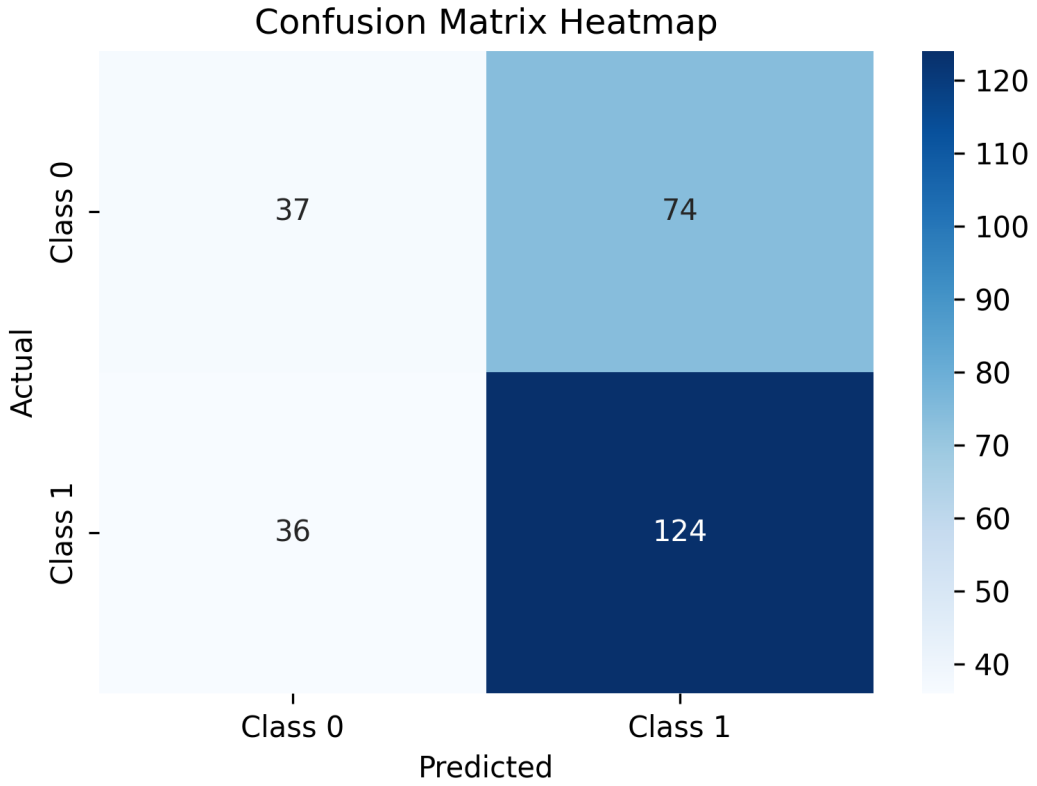

Figure 3.1: Confusion Matrix Heatmap.

3.5.3 Challenges and Limitations

- **Data Quality and Availability** The accuracy of predictions heavily depends on the quality of news data and the reliability of sentiment scores. Erroneous or overly generalized sentiment scores may lead to incorrect predictions.
- **Temporal Misalignment** Aligning news articles with stock price movements presented challenges due to delays in market reactions and differences in data granularity. For instance, intraday price movements could not always be correlated with daily news sentiment.
- **Complex Market Dynamics** Stock prices are influenced by a myriad of factors beyond news sentiment, such as macroeconomic indicators, geopolitical events, and trader psychology. This introduces noise that the model cannot fully capture.
- **Overfitting Risk** The model's high complexity and sensitivity to input features increased the risk of overfitting, especially on small or highly specific datasets.

Chapter 4

Conclusion

The study, titled *Natural Language Processing and Stock Prediction Using News Sentiments*, explores the integration of natural language processing (NLP) and deep learning techniques to predict stock market movements using financial news and historical stock prices. This research aimed to leverage sentiment analysis from news articles to complement historical price data, thereby enhancing the predictive power of machine learning models in financial markets.

1. **Summary of Contributions**

    - **News Sentiment as a Key Predictor** By extracting sentiment scores from financial news articles, the project demonstrated that textual data contains valuable signals about stock market trends. Positive and negative sentiments were often found to correlate with stock price increases and decreases, respectively.

    - **Sequential Modeling with LSTM** The Long Short-Term Memory (LSTM) network proved effective in capturing sequential dependencies within the dataset. Its ability to process time-series data and handle long-term dependencies was instrumental in modeling the intricate relationships between news sentiments, stock prices, and future movements.

    - **Hybrid Feature Set** The integration of both news sentiment and historical stock prices into the predictive framework highlighted the complementary nature of these datasets. The hybrid approach outperformed models relying solely on either news data or stock prices.

    - **Automated Workflow** The project established an automated pipeline for data collection, preprocessing, feature engineering, and model training. This workflow can be adapted for real-time predictions, paving the way for practical applications in algorithmic trading.

2. **Key Insights**

    - **Complex Interdependencies** Stock prices are influenced by a wide range of factors beyond news sentiment, including macroeconomic indicators, global market

trends, and geopolitical events. While this project focused on news-driven sentiment analysis, the broader market dynamics add layers of complexity that must be accounted for in future models.

- **Importance of Data Alignment** Aligning news data with corresponding stock price movements is crucial for accurate predictions. Temporal mismatches can reduce model effectiveness and introduce noise into the system.
- **Model Flexibility** The LSTM model's flexibility to adapt to various inputs (text embeddings, numerical data) demonstrates its suitability for financial prediction tasks. However, the risk of overfitting underscores the need for careful hyperparameter tuning and validation.

3. **Ethical Considerations Market Manipulation Risks** Addressing the potential risks of market manipulation and biases in prediction models will be crucial for responsible deployment in the financial domain.

   This study lays a foundation for leveraging NLP and deep learning techniques in financial market prediction, highlighting both the potential and challenges of integrating diverse data sources for informed decision-making.

4.1 Limitations

Despite its promising results, the project faced certain limitations:
Data Bias and Quality Sentiment scores derived from generic NLP models may not fully capture the nuances of financial news. Domain-specific sentiment models could yield better results.
Lack of External Features The model did not incorporate external economic indicators or global market data, which could further improve predictions.
Scalability to Real-Time Systems While the current pipeline provides batch predictions, its scalability to real-time applications remains unexplored.

4.2 Future Directions

Advanced Architectures Future studies can explore Transformer-based models like BERT or GPT, which are better suited for understanding contextual nuances in text data.
Real-Time Systems Implementing a real-time prediction system with live news feeds and stock prices would enhance the practical utility of this approach.
Expanded Feature Set Incorporating macroeconomic indicators, industry-specific variables, and social media sentiment could make predictions more robust and comprehensive.

- Challenges in Sentiment Analysis: Despite advancements, challenges remain in accurately capturing sentiment nuances, particularly in financial jargon and ambiguous language. Misclassifications can lead to erroneous predictions, highlighting the need for improved models.
- Gaps in Research: While many studies focus on specific stocks or sectors, there is a need

for comprehensive research that explores the generalizability of findings across different industries and market conditions. Additionally, integrating alternative data sources, such as economic indicators and trading volume, alongside sentiment analysis could enhance predictive models.

Appendix A

Appendix

A.1 Data Extraction

A.1.1 Stock Data Collection

```
# Get stock data for Reliance Industries
stock_symbol = 'RELIANCE.NS'
stock_data = yf.Ticker(stock_symbol)

# Download stock data for the last year
stock_hist = stock_data.history(period='1mo')

# Keep only the Date and Close columns
stock_hist = stock_hist[['Close']]
stock_hist.reset_index(inplace=True)


import pandas as pd
from datetime import datetime
stock_hist['Date'] = pd.to_datetime(stock_hist['Date'])
stock_hist['Date'] = stock_hist['Date'].dt.date
# Print the first few rows of stock data
print(stock_hist)
```

Listing A.1: Python code for stock data collection

A.1.2 Stock News Collection

```

import pandas as pd
from newsapi import NewsApiClient

newsapi = NewsApiClient(api_key='3e6aec89330749dea5a93b37487eff32')
```

```

# Collect news about Reliance Industries from 1st Jan 2022 to 15th
    Sep 2024
news = newsapi.get_everything(
    q='Reliance Industries',
    from_param='2024-09-08',
    to='2024-10-05',
    language='en',
    sort_by='relevancy',
)

# Create a DataFrame from the news articles
news_data = pd.DataFrame(news['articles'])
news_data['publishedAt'] =
    pd.to_datetime(news_data['publishedAt']).dt.date

# Select relevant columns
news_data = news_data[['publishedAt', 'title', 'content', 'url']]
news_data.to_csv("Reliance_news_data2.csv")
# Print the first few rows of news data
print(news_data)
```

Listing A.2: Python code for stock news collection

A.1.3 Data Alignment

```
# Merge news and stock data
merged_data = pd.merge(news_data, stock_data, on='Date')
merged_data = merged_data.sort_values('Date')

# Feature selection
features = ['Sentiment_Score', 'Close']
X = merged_data[features]
y = merged_data['Movement']
```

Listing A.3: Python code for aligning stock data and news data

A.2 Data Preprocessing

A.2.1 Lower Casing

```
#1. Lower_Casing
import pandas as pd
data=pd.read_csv(r"C:\Users\marat\OneDrive\Desktop\2nd
    year\project\Reliance_news_data2.csv")
df=data
```

```
df['content'] = df['content'].astype(str)
df['content']=df['content'].apply(lambda x: x.lower())
print(df.head())
```

Listing A.4: Python code for lower casing

A.2.2 Tokenization

```
#2. Remove_Punctuations

import string
string.punctuation

exclude=string.punctuation

def rem_pun(text):
    for char in exclude:
        text=text.replace(char,'')
    return text

import re
def remove_tags(raw_text):
    cle_text=re.sub(re.compile('<.*?>'),'',raw_text)
    return cle_text

df['content']=df['content'].apply(rem_pun)
print(df.head())
```

Listing A.5: Python code for tokenization

A.2.3 Stemming

```
#4. Stemming
from nltk.stem.porter import PorterStemmer


def stem_tokens(tokens):
  stemmer = PorterStemmer()
  stemmed_tokens = [stemmer.stem(token) for token in tokens]
  return stemmed_tokens

def stem_dataframe(df):
  df['stemmed_contents'] =
     df['tokenized_contents'].apply(stem_tokens)
  return df

# Stem the tokens
```

```
df = stem_dataframe(df)
print(df.head())
```

Listing A.6: Python code for stemming

A.2.4 Lemmatization

```


import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet


def get_wordnet_pos(word):
  tag = nltk.pos_tag([word])[0][1][0].upper()
  tag_dict = {"J": wordnet.ADJ,
              "N": wordnet.NOUN,
              "V": wordnet.VERB,
              "R": wordnet.ADV}
  return tag_dict.get(tag, wordnet.NOUN)

def lemmatize_tokens(tokens):
  lemmatizer = WordNetLemmatizer()
  lemmatized_tokens = [lemmatizer.lemmatize(token,
     get_wordnet_pos(token)) for token in tokens]
  return lemmatized_tokens

def lemmatize_dataframe(df):
  df['lemmatized_contents'] =
     df['tokenized_contents'].apply(lemmatize_tokens)
  return df

# Lemmatize the tokens
df = lemmatize_dataframe(df)
print(df.head())
```

Listing A.7: Python code for lemmatization

A.3 Text Representation

A.3.1 One Hot Encoding

```
#1. one hot encoding
from sklearn.preprocessing import OneHotEncoder
import itertools
```

```
4
5  tokens = [doc.split(" ") for doc in df.iloc[1,6]]
6
7  token_chain = itertools.chain.from_iterable(tokens)
8  word_to_id = {token: idx for idx, token in
       enumerate(set(token_chain))}
9  token_ids = [[word_to_id[token] for token in toke] for toke in
       tokens]
10
11 vec = OneHotEncoder(categories="auto")
12 V = vec.fit_transform(token_ids)
13 print(df.head())
14 print(V.toarray())
```

Listing A.8: Python code for One Hot Encoding

A.3.2 Bag of Words

```
1
2  import sklearn
3  from sklearn.feature_extraction.text import CountVectorizer
4
5
6
7  cv = CountVectorizer()
8  bow = cv.fit_transform(combined_list)  # Replace with df.iloc[1, 8]
       as per your original code
9
10 # Create a DataFrame from the BoW matrix
11 bow_df = pd.DataFrame(bow.toarray(),
       columns=cv.get_feature_names_out())
12
13 # Vocabulary
14 vocab = cv.vocabulary_
15
16 # Print Vocabulary
17 print("Vocabulary:")
18 print(vocab)
19
20 # Display the Bag of Words DataFrame
21 print("\nBag of Words DataFrame:")
22 print(bow_df)
23
24 vocab_summary = pd.DataFrame([vocab]).rename(index={0: 'Vocabulary'})
25
26 final_output = pd.concat([vocab_summary, bow_df], ignore_index=False)
27
28 print("\nFinal Output (BoW DataFrame with Vocabulary):")
```

```
print(final_output)
```

Listing A.9: Python code for Bag of Words

A.3.3 TF-IDF

```
#3. Term Frequency and Inverse Document Frequency (TF-IDF)
from sklearn.feature_extraction.text import TfidfVectorizer

tf = TfidfVectorizer()
txt_fit = tf.fit(df.iloc[1,6])
txt_transform = txt_fit.transform(df.iloc[1,6])
idf = tf.idf_
print(dict(zip(txt_fit.get_feature_names_out(), idf)))

#
tokenized_sentences = df['tokenized_contents'].tolist()
txt_fit = tf.fit(combined_list)
txt_transform = txt_fit.transform(combined_list)
idf = tf.idf_
print(dict(zip(txt_fit.get_feature_names_out(), idf)))
```

Listing A.10: Python code for TF-IDF

A.3.4 Word2Vec

```
import pandas as pd
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk


model = Word2Vec(sentences=tokenized_sentences, vector_size=100,
    window=5, min_count=1, workers=4)

# Example: Get the vector for a specific word
word_vector = model.wv['reliance']  # Replace 'document' with the
    word you want to check
print("Vector for 'reliance':")
print(word_vector)

# Example: Finding similar words
similar_words = model.wv.most_similar('reliance', topn=5)
print("\nMost similar words to 'reliance':")
print(similar_words)
```

Listing A.11: Python code for Word2Vec

A.3.5 Scaling of Features

```

# Scale features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Reshape data for LSTM
sequence_length = 10  # Use 10 days of data to predict movement
X_sequences = []
y_sequences = []

for i in range(sequence_length, len(X_scaled)):
    X_sequences.append(X_scaled[i-sequence_length:i])
    y_sequences.append(y.values[i])

X_sequences = np.array(X_sequences)
y_sequences = np.array(y_sequences)
```

Listing A.12: Python code for scaling of features

A.4 Modelling

A.4.1 Calculation of Sentiment Score

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import pandas as pd
import nltk

# Download VADER lexicon if not already downloaded
#nltk.download('vader_lexicon')

# Initialize the SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

# Calculate sentiment scores
news_data["Sentiment_Score"] = news_data["News"].apply(lambda x:
   sia.polarity_scores(x)["compound"])

# Display the dataframe with sentiment scores
news_data.head()
```

Listing A.13: Python code for calculation of sentiment score

A.4.2 LSTM Model

```
# Build the LSTM model
from tensorflow.keras.layers import BatchNormalization

model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(X_train.shape[1],
       X_train.shape[2])),
    Dropout(0.2),
    BatchNormalization(),
    LSTM(64),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy',
   metrics=['accuracy'])


class_weight = {0: 1.5, 1: 1}  # Adjust the weight for class 0 to be
   higher
history =model.fit(X_train, y_train, epochs=20, batch_size=32,
   validation_data=(X_test, y_test), class_weight=class_weight)
```

Listing A.14: Python code for LSTM model

A.4.3 Evaluation

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay



# Plot heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
   xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0',
   'Class 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.savefig(r'C:\Users\marat\OneDrive\Desktop\2nd
   year\project\confmat.png', format='png', dpi=300)
plt.show()
```

Listing A.15: Python code for evaluation

A.4.4 Evaluation Metrics

```
# Predict

from sklearn.metrics import classification_report, confusion_matrix,
    accuracy_score, precision_score, recall_score, f1_score

conf_matrix=confusion_matrix(y_test, y_pred)
# Evaluation metrics
print("Confusion Matrix:")

print(confusion_matrix(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"Precision: {precision_score(y_test, y_pred)}")
print(f"Recall: {recall_score(y_test, y_pred)}")
print(f"F1 Score: {f1_score(y_test, y_pred)}")
```

Listing A.16: Python code for evaluation metrics

—

—

—

A.5 Hardware and Software Specifications

The project was implemented using the following specifications:

- **Hardware:**
  - Processor: Ryzen 5, 3.4 GHz
  - RAM: 16 GB
  - GPU: NVIDIA GEFORCE GTX
- **Software:**
  - Programming Language: Python 3.10
  - IDE: Jupyter Notebook
  - Libraries: TensorFlow, NLTK, SpaCy, scikit-learn, pandas

The jupyter notebook with all codes and the dataset used for building model are available at " Github/Harshal Marathe".

Bibliography

Heyan Huang, Xiao Liu, Yue Zhang, and Chong Feng. News-driven stock prediction via noisy equity state representation. *Neurocomputing*, 470:66–75, 2022.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson, 3rd edition, 2023.

Paul C Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The Journal of finance*, 62(3):1139–1168, 2007.

Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(5): 1–28, 2018.